Design and Analysis of Algorithms Assignment 2

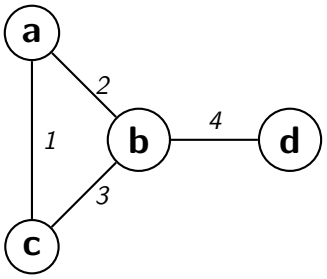
Harrison Lee, Alex Zhao

February 1, 2019

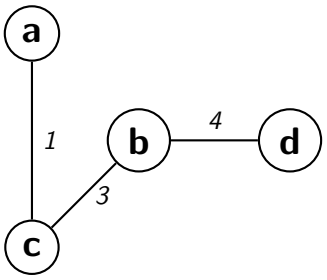
Question 1. *(4.9)*

a. *A MBST is not always a MST*

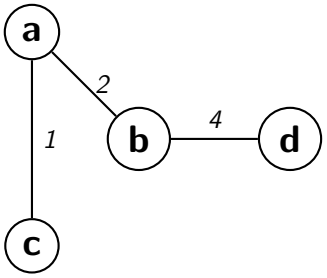
Given G:



A MBST of G is:



The MST of G is:



b. *A MST is always a MBST*

Let T be the MST of graph G. Assume that T is not the MBST T' of G. That means that there is a bottleneck edge e in T not in T' that is heavier than the bottleneck edge b of T' (all edge weights are distinct). This means that e is heavier than every edge in T'. By adding e to T', we create a cycle where e is the heaviest edge in this cycle. By the cycle property, this edge cannot be part of any MST. This is a contradiction. Therefore, our assumption was false and T must be a MBST.

Question 2. *(6.11) Dynamic Programming*

Given: You have n weeks and s_i parts to be produced each of those weeks. You must decide between two shipping companies, Company A which charges $r * s$ to ship in a given week while Company B charges c each week in blocks of four consecutive weeks.

Find: A schedule deciding between company A or B for each of those n weeks while following company B's restrictions. Cost is the amount paid in shipping costs.

Give a polynomial time algorithm that takes a sequence of supply values, $s_1, s_2, \ldots, s_n$ and returns a schedule of minimum cost.

Algorithm 1. *Find the optimal schedule.*

Proof. Subproblems: $OPT(j)$, or the optimal schedule from week 0 to week j. This can be expanded from week 4 to week n.

Recurrence: $OPT(j) = min\{^{r * s_j + OPT(j-1)}_{4 * c + OPT(j-4)}$, where $OPT(j)$ represents the optimal shipping costs possible from week 0 to week j. r, s, and c are as defined in the problem, referring to company A's cost per weight unit, total weight during week j, and company B's cost per week respectively.

Full Algorithm:

OPT[0, 1, 2, 3] = $[0, s_1 * r, s_1 * r + s_2 * r, s_1 * r + s_2 * r + s_3 * r]$

for j = 0, j $\leq$ n, j++

  if $OPT[j-4] + 4 * c < OPT[j-1] + r * s_j$

    $OPT[j] = OPT[j-4] + 4 * c$

  else:

    $OPT[j] = OPT[j-1] + r * s_j$

$return OPT[n]$

Running Time: $O(n)$. Each week is traversed once, while two already calculated $OPT(j)$s are accessed, $OPT(j-1)$ and $OPT(j-4)$, each week.

□